




**Department of
Aerospace Engineering**
Faculty of Engineering
& Architectural Science

| | |
|------------------------------|---|
| Semester (Term, Year) | Fall, 2023 |
| Course Code | AER850 |
| Course Section | 1 |
| Course Title | Introduction to Machine Learning |
| Course Instructor | Dr. Reza Faieghi |
| Submission | Project Report |
| Submission No. | 1 |
| Submission Due Date | 10/15/2023 |
| Title | Project 1: Machine Learning |
| Submission Date | 10/15/2023 |

| | | |
|------------------------------|------------------------------|---|
| Submission by (Name): | Student ID (XXXX1234) | Signature |
| Mehtab Singh | XXXX60754 |  |

By signing the above you attest that you have contributed to this submission and confirm that all work you contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, and "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Academic Integrity Policy 60, which can be found at www.torontomu.ca/senate/policies/

Aerospace Assignment Cover as of May 2022

Table of Contents

| | |
|--|-----------|
| 1. Introduction and Project Outline | 3 |
| 2. Results & Discussion | 4 |
| 2.1 - Data Processing and Visualization | 4 |
| 2.2 - Correlation Analysis | 6 |
| 2.3 - Classification Model Development/Engineering | 8 |
| 2.3.1- Model 1: Logistic Regression | 8 |
| 2.3.2- Model 2: Random Forest Regressor | 8 |
| 2.3.3- Model 3: Decision Tree | 9 |
| 2.4 - Model Performance Analysis and Evaluation | 9 |
| 2.4.1- Model Performance Analysis | 10 |
| 2.4.2- Model Performance Evaluation | 11 |
| 3. Conclusion | 12 |
| References | 13 |
| Appendix | 14 |

List of Figures

| | |
|---|----|
| Figure 1: FlightMax Fill Motion Simulator Inverter | 3 |
| Figure 2: 3D Plot for the Motion Simulator Step Measurements | 4 |
| Figure 3: The Step Plot for the X-Direction | 5 |
| Figure 4: The Step Plot for the Y-Direction | 5 |
| Figure 5: The Step Plot for the Z-Direction | 6 |
| Figure 6: The Heatmap Plot for the Correlation Matrix | 7 |
| Figure 7: Confusion Matrix from Logistic Regression Classification | 10 |
| Figure 8: Confusion Matrix from Decision Tree Classification | 11 |

List of Tables

| | |
|---|----|
| Table 1: The Correlation Values with respect to the Step Array | 7 |
| Table 2: The Classification Report for the Logistic Regression Model | 8 |
| Table 3: The Classification Report for the Random Forest Regressor Model | 9 |
| Table 4: The Classification Report for the Decision Tree Model | 9 |
| Table 5: Predicted Step Values | 11 |

1. Introduction and Project Outline

The project focuses on the analysis of the data provided by the disassembly of the inverter below.



Figure 1: FlightMax Fill Motion Simulator Inverter [1]

13 steps provide X, Y, and Z coordinates that are provided.

2. Results & Discussion

The following are the steps taken in the project, the code for which is uploaded in the GitHub repository which is linked in the Appendix.

2.1 - Data Processing and Visualization

To start off, the data was plotted in different ways starting with the 3D point of view for the axis points all the way through to each point as per the step value. Using the Numpy and Matplotlib packages yielded the results below.

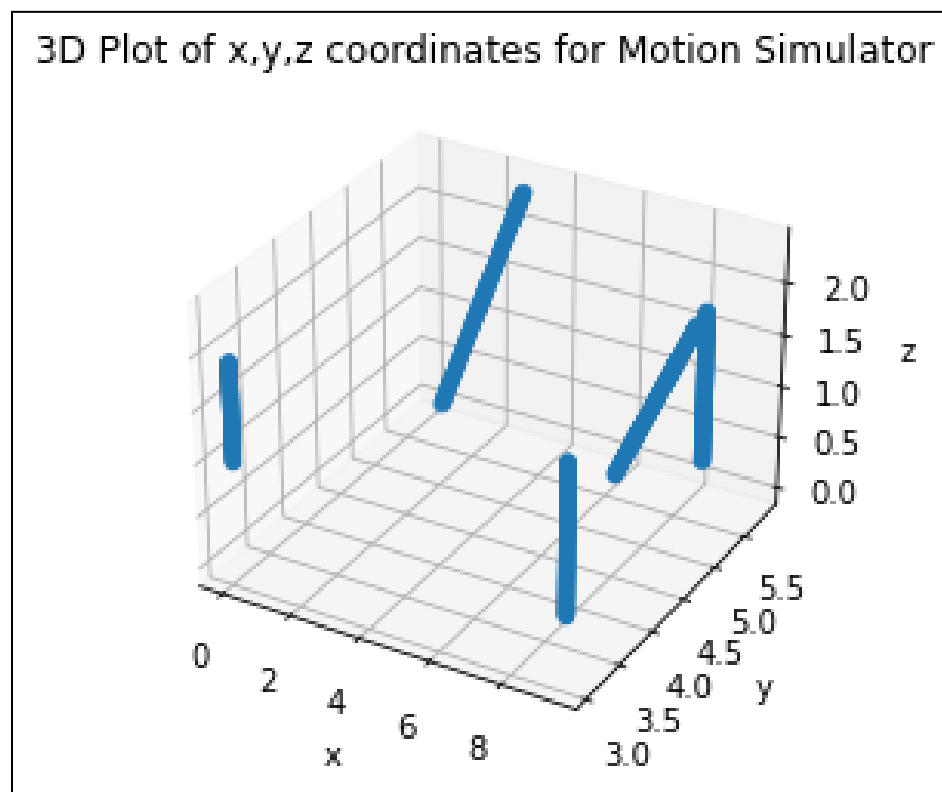


Figure 2: 3D Plot for the Motion Simulator Step Measurements

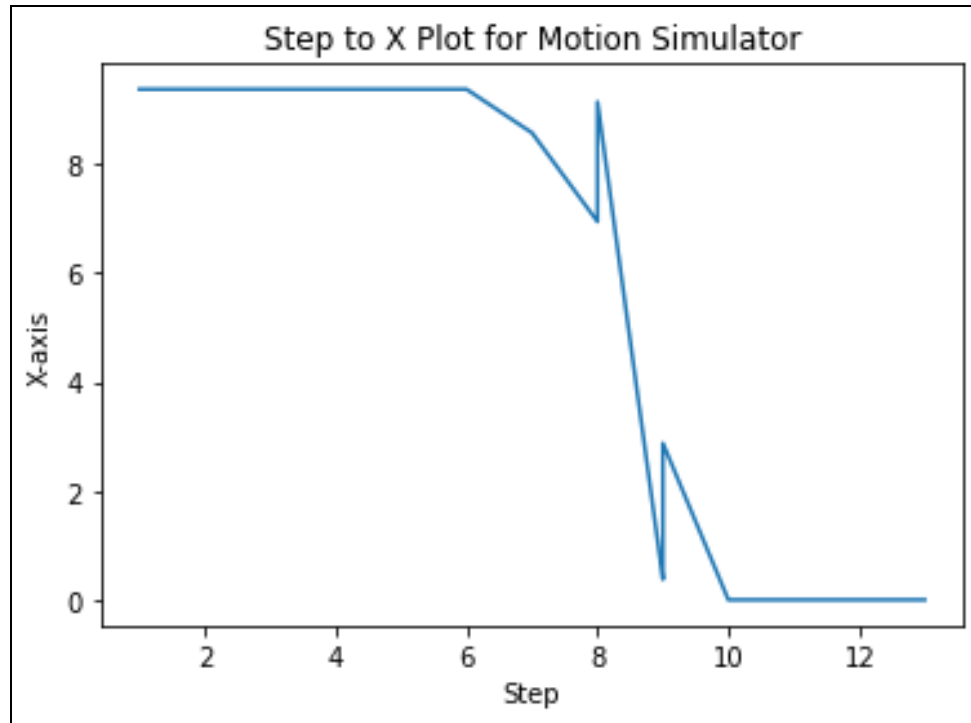


Figure 3: The Step Plot for the X-Direction

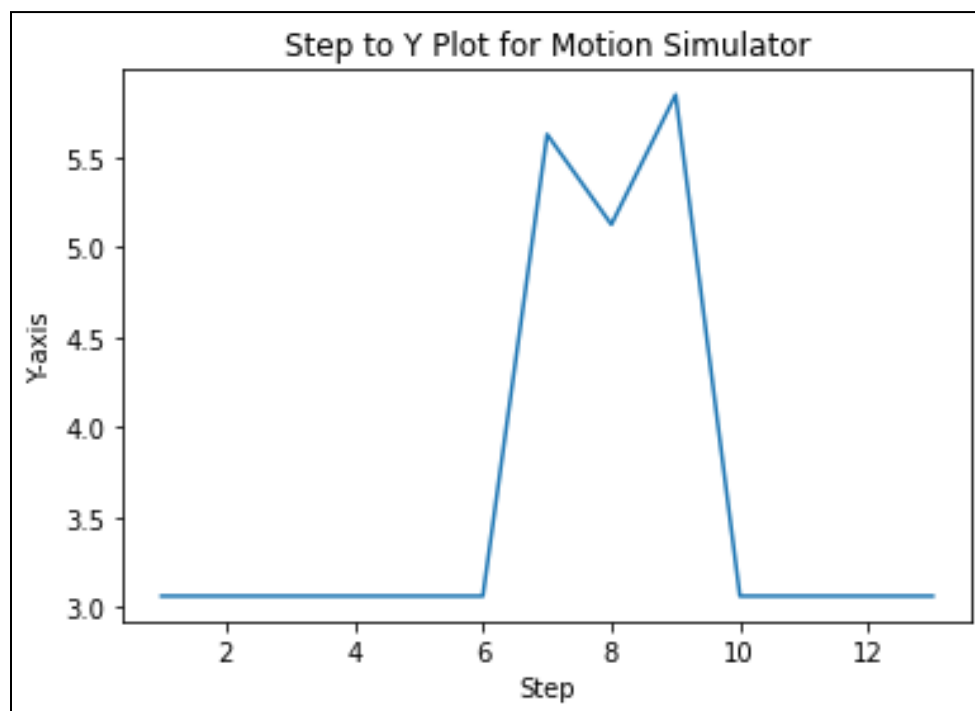


Figure 4: The Step Plot for the Y-Direction

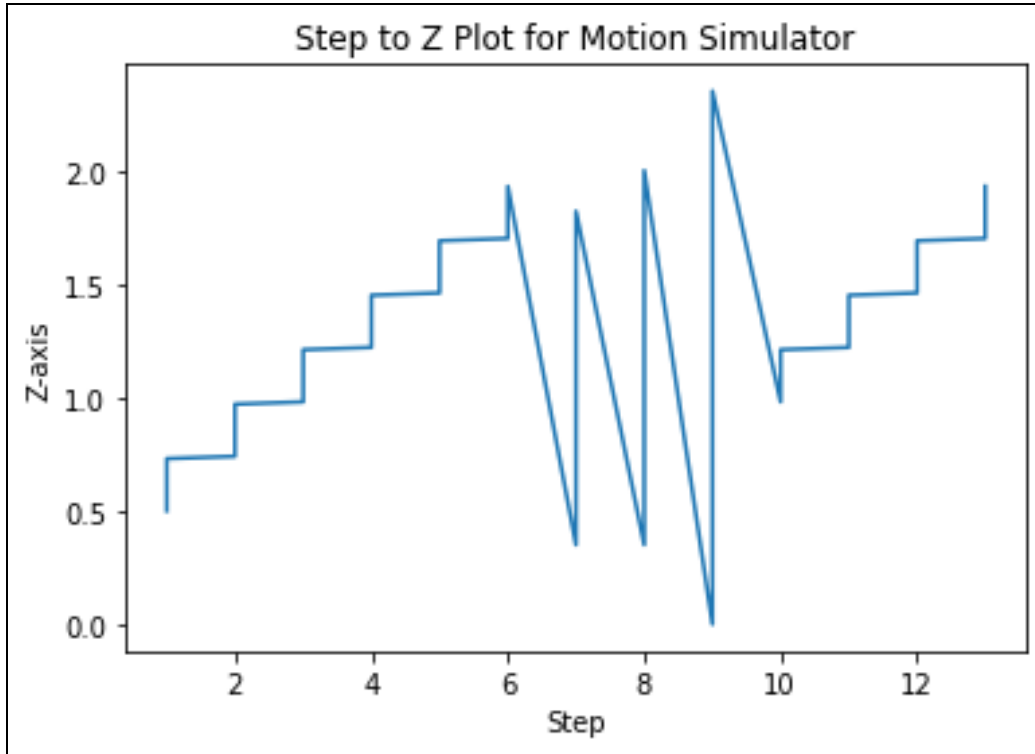


Figure 5: The Step Plot for the Z-Direction

In the 3D plot figure 2 it showed grouped values of straight lines, meaning at different points of Z the X and Y were not changing greatly. When looking at the figures 3 to 5 they showed varying correlations, but one similarity was that the values seemed to be skewed from step 6 to 10 for all axis points. Looking past step 6 to 10, there is a constant change in each of the coordinate axis points. In figure 3, the x-coordinates showed a decline from 9 to 0 as each step continued. In figure 4, the y-coordinates showed minimal change staying above 3 as each step continued. In figure 5, the z-coordinates showed a fluctuating set of values going from 0.5 to 1.75 and back to 1.0 to finally 2.0. Knowing this, the correlation is what needs to be visualized to help with the classification of models.

2.2 - Correlation Analysis

The correlation analysis was done using a heatmap plot, the values for which were found using the stratified sampling method from the *sklearn.model_selection* package. This method takes the base data within the *dataframe* and splits it into its separate components X, Y, Z, and Step. From there Step was the *stratify* variable as it had the most effect on the rest of the components. As such X, Y, and Z can be compared to the Step to find the correlation using the *corrcoef()[i,j]* function. The heatmap can also be plotted using a similar method, starting with the *dataframe.corr()* function to create a correlation matrix from the original *dataframe*. From there

the heatmap can be plotted using the *seaborn* library and the `.heatmap()` command. The following below are the results from both the plot and correlation values.

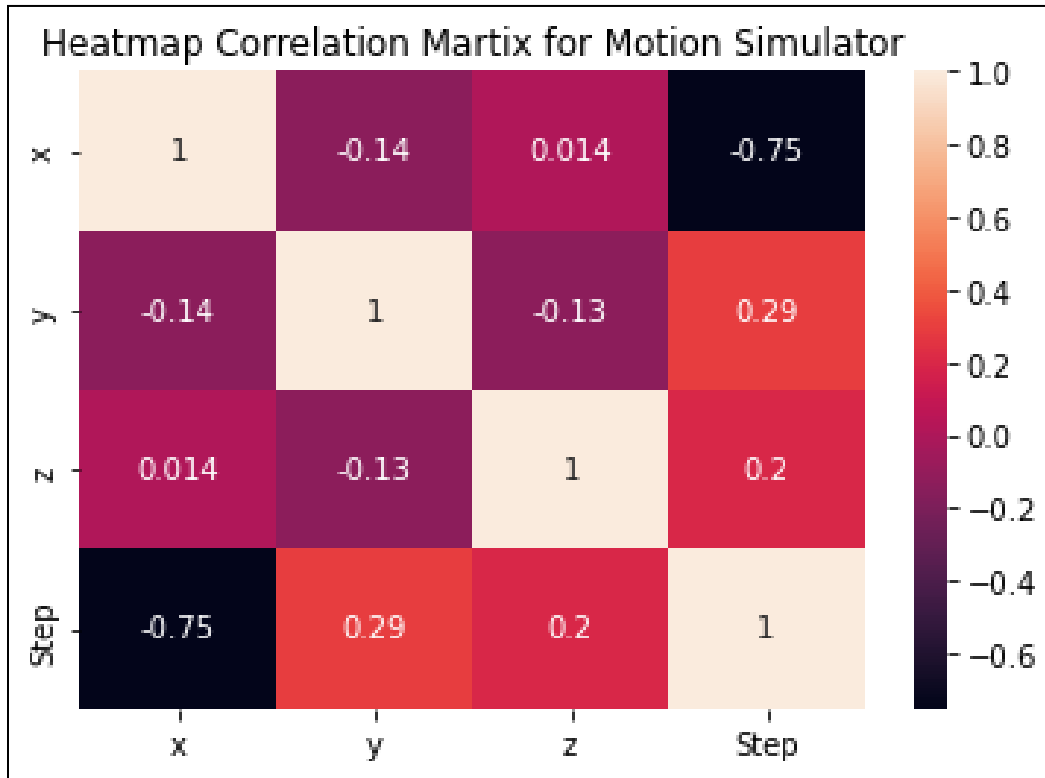


Figure 6: The Heatmap Plot for the Correlation Matrix

Table 1: The Correlation Values with respect to the Step Array

| Correlation with Step: | |
|------------------------|-----------|
| Y | 0.294097 |
| X | -0.749724 |
| Z | 0.192925 |
| step | 1.000000 |

Considering that the *Step* is the one to be compared with, it makes sense to have the correlation be 1.0. As for the *X* set, it had the least correlation of -0.75 which could have been due to the skewed *Step* from 6 to 10 that was presented in figure 3. *Y* set had 0.29 as it partially stayed consistent at around 3 before spiking to 5.5 and back down. *Z* set had 0.19 as it was an increasing slope that shifted during *Step* 6 to 10. The heatmap presents the full range of correlations. When ranking the best to worst <1 correlation values, they are: *Step-Y*, *Step-Z*, *X-Z*, *Y-Z*, *X-Y*, & *Step-X*. This helps visualize the best areas the program will focus on when creating a classification model.

2.3 - Classification Model Development/Engineering

Three models are used to form classification development, the following are used below.

2.3.1- Model 1: Logistic Regression

The Logistic Regression method defines the probability of whether or not the chosen event will occur. As such using it with multiple data sets with varying data could hinder its accuracy. Using the *LogisticRegression()* and *.fit()* functions helped to train the data when using stratified sampling from 2.2. Taking the sets *X*, *Y*, *Z* as the *X_train* and *X-test* 2D array while the *Step* set was denoted *y_train* and *y_test* array considering the *Step* presents the main comparison value. From there the *.predict()* command uses the *X_test* to get the predicted values for the *Step* array *y_pred*. This offers comparison to the predicted and actual *Step* values, thus yielding the table below.

Table 2: The Classification Report for the Logistic Regression Model

| Accuracy: 0.8488372093023255 | | | | |
|------------------------------|-----------|--------|----------|---------|
| Classification Report: | | | | |
| | precision | recall | f1-score | support |
| 1 | 0.38 | 1.00 | 0.55 | 3 |
| 2 | 0.00 | 0.00 | 0.00 | 5 |
| 3 | 0.43 | 1.00 | 0.60 | 6 |
| 4 | 0.00 | 0.00 | 0.00 | 8 |
| 5 | 1.00 | 0.40 | 0.57 | 5 |
| 6 | 0.57 | 1.00 | 0.73 | 4 |
| 7 | 1.00 | 0.73 | 0.84 | 26 |
| 8 | 0.86 | 1.00 | 0.92 | 43 |
| 9 | 1.00 | 1.00 | 1.00 | 52 |
| 10 | 0.50 | 1.00 | 0.67 | 3 |
| 11 | 1.00 | 0.25 | 0.40 | 4 |
| 12 | 1.00 | 1.00 | 1.00 | 6 |
| 13 | 1.00 | 1.00 | 1.00 | 7 |
| accuracy | | | 0.85 | 172 |
| macro avg | 0.67 | 0.72 | 0.64 | 172 |
| weighted avg | 0.84 | 0.85 | 0.82 | 172 |

The accuracy of the classification ended up being 85% which is good but gives way to erroneous predictions.

2.3.2- Model 2: Random Forest Regressor

Same like the Logistic Regression, the Random Forest Regressor presents a similar form to obtain the classification values. The values for the MAE of which are presented below.

Table 3: The Classification Report for the Random Forest Regressor Model

```

Mean Squared Error: 0.0195796511627907
Mean Absolute Error: 0.02994186046511628
R-squared (R2): 0.996812908053503

```

The previous and next model are of interest as those present the prediction accuracy for each and every *Step* value compared to a broad average.

2.3.3- Model 3: Decision Tree

The Decision Tree uses *DecisionTreeClassifier()* to present a list of paths for the data set predictions, as such it has more potential to be accurate in scenarios with long sets of varying data. When utilizing the same method done in section 2.3.1, the following accuracy was found below.

Table 4: The Classification Report for the Decision Tree Model

```

Accuracy: 0.9883720930232558
Classification Report:

```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 1 | 1.00 | 1.00 | 1.00 | 3 |
| 2 | 1.00 | 1.00 | 1.00 | 5 |
| 3 | 1.00 | 1.00 | 1.00 | 6 |
| 4 | 1.00 | 1.00 | 1.00 | 8 |
| 5 | 0.83 | 1.00 | 0.91 | 5 |
| 6 | 1.00 | 0.75 | 0.86 | 4 |
| 7 | 1.00 | 1.00 | 1.00 | 26 |
| 8 | 1.00 | 1.00 | 1.00 | 43 |
| 9 | 1.00 | 1.00 | 1.00 | 52 |
| 10 | 1.00 | 1.00 | 1.00 | 3 |
| 11 | 1.00 | 1.00 | 1.00 | 4 |
| 12 | 0.86 | 1.00 | 0.92 | 6 |
| 13 | 1.00 | 0.86 | 0.92 | 7 |
| accuracy | | | 0.99 | 172 |
| macro avg | 0.98 | 0.97 | 0.97 | 172 |
| weighted avg | 0.99 | 0.99 | 0.99 | 172 |

Compared to the Logistic Regression, this model is more accurate being 98%. As such it will be the model to be evaluated in section 2.4.2.

2.4 - Model Performance Analysis and Evaluation

As the model first needs to be plotted, it is best to plot both the Logistic Regression and Decision Tree Classification Models to compare and contrast the best model.

2.4.1- Model Performance Analysis

The heatmap `.heatmap()` command is used to plot both Classification Models as a Confusion Matrix. A Confusion Matrix is meant to show the predicted and actual values– the more diagonal the values are grouped together, the more accurate the model is. Below are the two Confusion Matrix models.

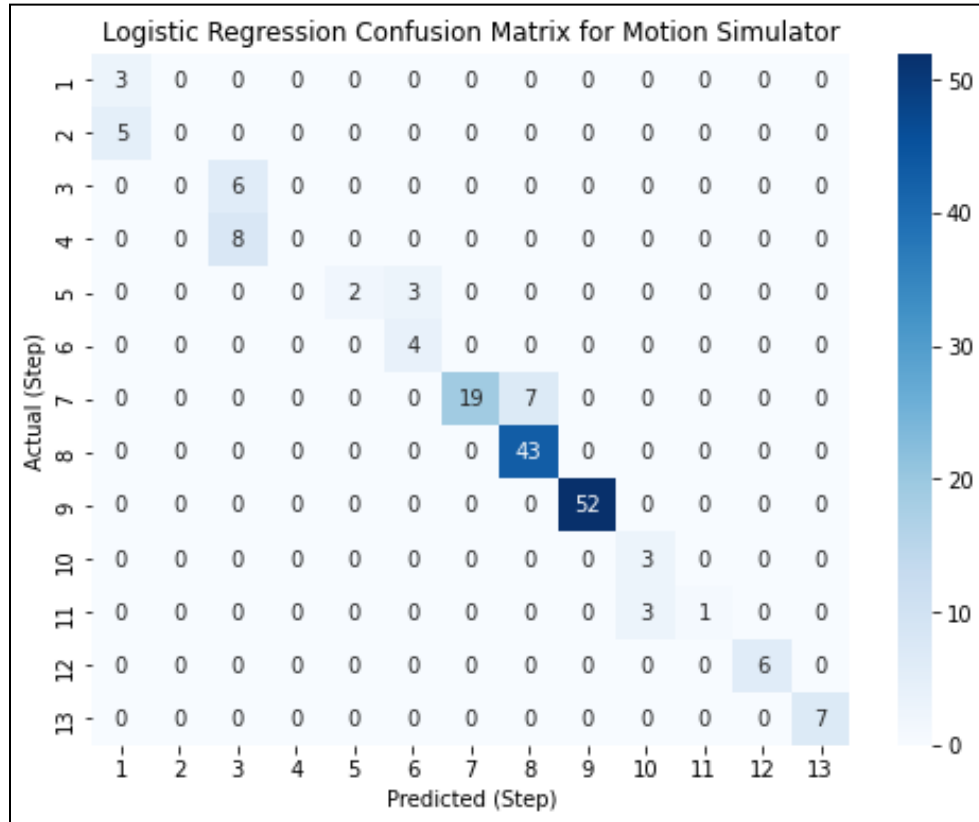


Figure 7: Confusion Matrix from Logistic Regression Classification

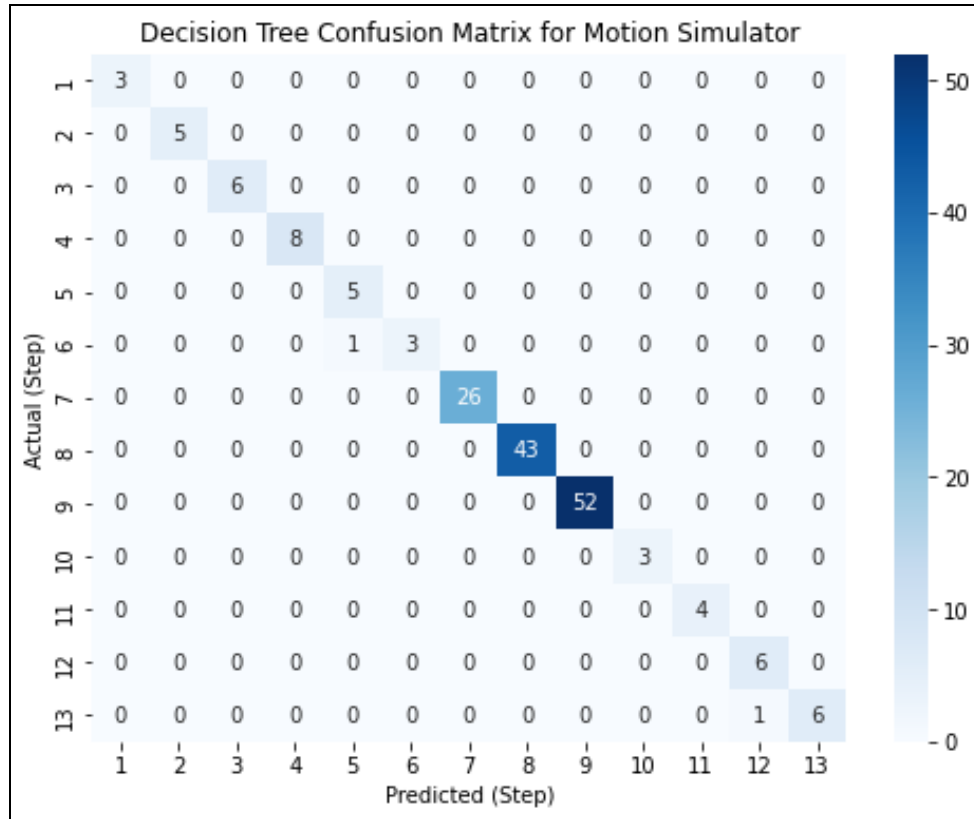


Figure 8: Confusion Matrix from Decision Tree Classification

Figure 8, Decision Tree, presented the most accurate results. As such, it will be used for the evaluation of the whole machine learning algorithm.

2.4.2- Model Performance Evaluation

The following data sets (X,Y,Z) below are used to evaluate the model performance:
 [9.375,3.0625,1.51], [6.995,5.125,0.3875], [0,3.0625,1.93], [9.4,3,1.8], [9.4,3,1.3]
 Using these X, Y, and Z axis points, the model will predict the outcome step values for each set. The model that will be used is the Decision Tree classification as it was the most accurate of the 3 classification models analyzed. The *.predict()* command was used with the input being the 2D array holding all the evaluating data sets. The result of this is seen below.

Table 5: Predicted Step Values

[5 8 13 6 4]

This yields the predicted complete data sets (X,Y,Z,Step) below:
 [9.375,3.0625,1.51,**5**], [6.995,5.125,0.3875,**8**], [0,3.0625,1.93,**13**], [9.4,3,1.8,**6**], [9.4,3,1.3,**4**]
 When comparing this to the graphs in figures 2 to 5, these values approximately line up to the region the sample data was plotted on.

3. Conclusion

This project helped understand the fundamental concepts and steps to building a machine learning algorithm. Taking multiple classification models and comparing them to see the most accurate prediction model. Considering there is no true answer for the predicted *Step* values, the only way to confirm if the prediction was accurate is by checking the plots done earlier in both the report and code. Had there been a primary test that presents the actual *Step* values in the evaluation, the program could be fine-tuned even more to provide more accurate results.

References

- [1] Dr. Reza Faieghi “AER850 Project #1”
<https://courses.torontomu.ca/d2l/le/content/804341/viewContent/5372589/View> [Accessed 10/10/2023]
- [2] Dr. Reza Faieghi, “dronify/AER850-Fall23”.
<https://github.com/dronify/AER850-Fall23/blob/main/classes.py> [Accessed 10/10/2023]

Appendix

The following link is for the GitHub repository:

https://github.com/Mehtab0Singh/AER850_Project1_MehtabSingh.git